FHLF01 - Project

Johanna Gustafson & Erik Lundell

a

1 Introduction

In this report a lens system on a Mars rover subjected to extreme temperature conditions is investigated. The shifting temperatures can cause significant deformation in the lenses and thus distort pictures taken through the lens. Analyzing this system to take sufficient precautions is therefore crucial for capturing high quality images from the rover.

The system is approximated as a flat disk with the thickness 1 cm, as shown in figure 1. The two outermost lenses are curved as an ellipse with semi-axes 0.1 cm and 0.25 cm.

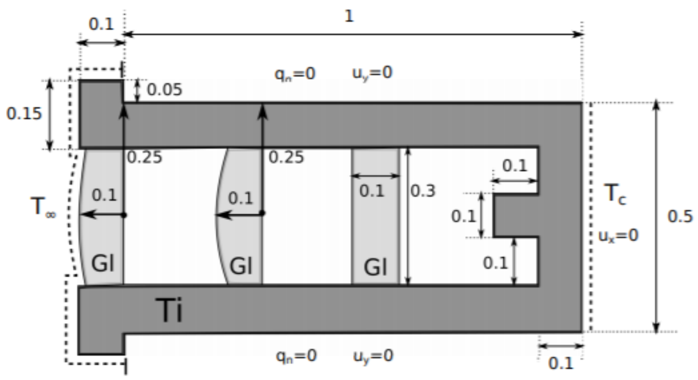


Figure 1: *The defined geometry of the lens system. All measurements in [cm] (image from project description).*

Different properties hold for the materials of the structure as shown in table 1.

		Titanium alloy	Glass
Young's modulus,	E [GPa]	110	67
Poisson's ratio,	ν [-]	0.34	0.2
Expansion coefficient,	α [1/K]	$9.4 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
Density,	ρ [kg/m^3]	4620	3860
Specific heat,	c_p [J/(kg K)]	523	670
Thermal conductivity,	k [W/(m K)]	17	0.8

Table 1: Material data.

The heat boundary condition for the inner boundaries is $q_n = 0$ as the spaces contain vacuum. The same condition holds for the top and bottom outer boundaries, whereas along the remaining outer boundaries there holds Newton convection; $q_n = \alpha_c(T - T_\infty)$ along the left side and $q_n = \alpha_c(T - T_c)$ along the right, where $\alpha_c = 100\,\mathrm{W/(m^2K)}$. The structure is stress-free at 20°C.

The lens is mounted such that the top and bottom boundaries are fixated in the y-direction, meaning $u_y = 0$. The rightmost boundary is fixated in the x-direction, meaning $u_x = 0$. Remaining constraint-free boundaries are traction free, meaning $\boldsymbol{t} = \boldsymbol{0}$. Plane strain conditions are assumed to hold.

Because of symmetry, only the top half of the system must be analyzed, as the solution of the bottom half is identical, but mirrored. The new boundary condition along the middle edge must be $q_n = 0$ and $u_y = 0$ to conserve symmetry.

The report concerns two base cases for the outer temperature T_∞. Day, where $T_\infty = 40$°C and night, where $T_\infty = -96$°C. The shift between the two temperatures is assumed to occur instantaneously.

2 Procedure

The mesh was defined using MATLAB's *pdetool* according to the specifications in figure 1. The output were the points $\boldsymbol{p}$, edges $\boldsymbol{e}$ and triangles $\boldsymbol{t}$, that together define the geometry and general material properties of the structure. These were saved in a file *data.mat*, which is loaded before any of the following steps.

The material properties are defined in the code of appendix A.1, according to table 1. Titanium is labeled 1 while glass is labeled 0 in order to automatically generate the correct constant for a certain element, using the method *containers.Map()*.

The $\boldsymbol{p}$, $\boldsymbol{e}$ and $\boldsymbol{t}$ matrices are used to define the elements and *edof*-matrix in CALFEM notation as seen in appendix A.2. Furthermore, edges are sorted according to their boundary condition. Edges are also labeled with either 1 or 0 in order to differentiate boundary conditions, in a similar fashion to that of material type.

2.1 The stationary temperature distribution problem

The Finite Element Formulation of the (2D) stationary temperature distribution problem is derived in Ottosen & Petersson (1992). The method is derived using the strong formulation of the heat equation,

$$\nabla \cdot \boldsymbol{q} = Q \tag{1}$$

and applying a constitutive law relating heat flux to temperature variations,

$$\boldsymbol{q} = -t\boldsymbol{D}\nabla T \tag{2}$$

where $\boldsymbol{D}$ is the constitutive matrix, Q denotes applied heat and t the disc thickness. The temperature is approximated as $T = \boldsymbol{Na}$, where $\boldsymbol{a}$ contains the nodal temperatures, meaning

$$\boldsymbol{q} = -t\boldsymbol{D}\nabla\boldsymbol{Na} = -t\boldsymbol{DBa}. \tag{3}$$

Equation 1 is transformed to its weak form on the region S with the boundary $\mathcal{L}$. The divergence theorem is used and equation 3 is applied where appropriate. Finally the Galerkin method is applied, where the weight function $\boldsymbol{v} = \boldsymbol{Nc}$ and $\boldsymbol{c}$ is arbitrary, to reach

$$\left(\int_S \boldsymbol{B}^T t\boldsymbol{DB}\, dS\right)\boldsymbol{a} = -\oint_{\mathcal{L}} \boldsymbol{N}^T q_n d\,\mathcal{L} + \int_S \boldsymbol{N}^T Qt\, dS, \tag{4}$$

where q_n is the heat flux normal to the boundary of the region. The value of q_n depends on the boundary condition.

The boundary conditions on $\mathcal{L}$ can consist of

- Dirichlet conditions (essential boundary conditions), where the temperature T is given.
- Neumann conditions (natural boundary conditions), where the heat flow q_n *from* the body is given.
- Convection, where q_n is given by Newton's heat equation: $q_n = \alpha(T - T_\infty)$.

As mentioned in the introduction, the left- and rightmost boundaries have convection while the top and bottom boundaries, as well as the inner boundaries, are insulated, where $q_n = 0$. The problem can be further simplified as the thickness t is constant, and as all materials are isomorphic so that $\boldsymbol{D} = k\boldsymbol{I}$. Finally $Q = 0$, resulting in the final formulation

$$t\int_A \boldsymbol{B}^T k\boldsymbol{B} dA\boldsymbol{a} = -\int_{\mathcal{L}_L} \boldsymbol{N}^T \alpha(T - T_\infty) d\mathcal{L} - \int_{\mathcal{L}_R} \boldsymbol{N}^T \alpha(T - T_0) d\mathcal{L}$$

$$\left(t\int_A \boldsymbol{B}^T k\boldsymbol{B} dA + \alpha\int_{\mathcal{L}_L+\mathcal{L}_R} \boldsymbol{N}^T\boldsymbol{N} d\mathcal{L}\right)\boldsymbol{a} = \alpha T_\infty \int_{\mathcal{L}_L} \boldsymbol{N}^T d\mathcal{L} + \alpha T_0 \int_{\mathcal{L}_R} \boldsymbol{N}^T d\mathcal{L} \tag{5}$$

$$\iff (\boldsymbol{K} + \boldsymbol{K_c})\boldsymbol{a} = \boldsymbol{f}_{b_L} + \boldsymbol{f}_{b_R} \tag{6}$$

where $\mathcal{L}_R$ refers to the outer right boundary and $\mathcal{L}_L$ the left. The notations $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{f}}$ are introduced as

$$\boldsymbol{K} + \boldsymbol{K_c} = \tilde{\boldsymbol{K}} \tag{7}$$

$$\boldsymbol{f_{b_L}} + \boldsymbol{f_{b_R}} = \tilde{\boldsymbol{f}} \tag{8}$$

to form the relation

$$\tilde{\boldsymbol{K}}\boldsymbol{a} = \tilde{\boldsymbol{f}}. \tag{9}$$

The boundary integrals of $\boldsymbol{K_c}$, $\boldsymbol{f_{b_L}}$ and $\boldsymbol{f_{b_R}}$ depend on the lengths of the edges along the boundaries. A single edge is located between the nearby nodes i and k, wherein the form functions N_i and N_k are non-zero, and the rest 0. As a result, it is reasonable to analyze two form functions at a time.

Between the nodes i and k, N_i and N_k take the form of linear functions, u_i and u_k respectively, for which $u_i : 1 \to 0$ and $u_k : 0 \to 1$. Letting $\mathcal{L}'$ denote the line between these nodes, and l its length, it is found that

$$\int_{\mathcal{L}'} [N_i \quad N_k]^T \, d\mathcal{L}' = \int_0^l [u_i \quad u_k]^T \, d\mathcal{L}' = [l/2 \quad l/2]^T, \text{ and} \tag{10}$$

$$\int_{\mathcal{L}'} [N_i \quad N_k]^T [N_i \quad N_k] \, d\mathcal{L}' = \int_0^l \begin{bmatrix} u_i^2 & u_i \cdot u_k \\ u_i \cdot u_k & u_k^2 \end{bmatrix} d\mathcal{L}' = \begin{bmatrix} l/3 & l/6 \\ l/6 & l/3 \end{bmatrix} \tag{11}$$

which are inserted into $\boldsymbol{K}_c$, $\boldsymbol{f}_{b_L}$ and $\boldsymbol{f}_{b_R}$ according to the nodal indices i and k.

2.1.1 MATLAB

The code solution of this problem is shown in appendix A.3. The stiffness matrix $\boldsymbol{K}$ of equation 5 is calculated elementwise using CALFEM's $flw2e.m$ to calculate the element stiffness matrices which are then assembled. $\boldsymbol{K}_c$ and $\tilde{\boldsymbol{f}}$ are calculated using equations 10 and 11 to then form equation 9. The linear equation system is then complete and it is possible to solve for the nodal temperatures $\boldsymbol{a}$ using CALFEM's *solveq.m*.

2.2 The transient temperature problem

The transient heat equation is similar to the stationary heat equation but contains an additional time derivative term,

$$\rho c_p \dot{T} + \nabla \cdot \boldsymbol{q} = Q. \tag{12}$$

Introducing $\dot{T} = \boldsymbol{N}\dot{\boldsymbol{a}}$, the FE formulation the derivative term in equation 12 becomes

$$\int_S \boldsymbol{N}^T \rho c_p \boldsymbol{N} \, dS \cdot \dot{\boldsymbol{a}} = \boldsymbol{A}\dot{\boldsymbol{a}} \tag{13}$$

resulting in the final equation

$$\tilde{\boldsymbol{K}}\boldsymbol{a} + \boldsymbol{A}\dot{\boldsymbol{a}} = \tilde{\boldsymbol{f}}, \tag{14}$$

which is solved using a time-stepping method. In this project the implicit Euler method is used to allow for longer time steps without risking divergence. With the notation $\boldsymbol{a}^n = \boldsymbol{a}(t_n)$, the chosen method is defined as

$$\dot{\boldsymbol{a}} =: \frac{\boldsymbol{a^{n+1}} - \boldsymbol{a^n}}{\Delta t} = g(\boldsymbol{a}^n), \tag{15}$$

where Δt is the length of the time step and g is a given function. Inserting this definition into equation 14 and solving for $\boldsymbol{a^{n+1}}$ yields the closed form

$$\boldsymbol{a^{n+1}} = \left(\boldsymbol{A} + \Delta t \tilde{\boldsymbol{K}}\right)^{-1}\left(\boldsymbol{A}\boldsymbol{a^n} + \Delta t \tilde{\boldsymbol{f}}\right). \tag{16}$$

2.2.1 MATLAB

This time-stepping method requires an initial value $\boldsymbol{a}(0) = \boldsymbol{a}_0$, which is produced by running the solution for the stationary heat problem (appendix A.3), setting T_∞ to either 20 °C or -96 °C depending on whether it initially is day- or nighttime. $\tilde{\boldsymbol{f}}$ is recalculated using the temperature condition opposite to the initial condition.

$\boldsymbol{A}$ is calculated elementwise using CALFEM's *plantml.m*. The number of time-steps are chosen so that $\boldsymbol{a}$ is calculated for t = 10, 100, 250 and 500 s.

2.3 The mechanical problem

Throughout the structure there are stresses present as a result of temperature variations and deformations. There are the normal stresses σ_{xx}, σ_{yy} and σ_{zz} in the x, y and z directions respectively, and the shear stresses τ_{jk} on the j plane, in the k direction. For the shear stresses it is found that $\tau_{xy} = \tau_{yx}$, $\tau_{xz} = \tau_{zx}$ and $\tau_{yz} = \tau_{zy}$ using moment equilibrium. Similar conditions hold for the normal strains ε and shear strains γ.

For linear thermoelasticity, the relation between an element's stress and tension

$$\boldsymbol{\sigma} = \begin{bmatrix}\sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \tau_{xy} & \tau_{xz} & \tau_{yz}\end{bmatrix}^T, \qquad \boldsymbol{\varepsilon} = \begin{bmatrix}\varepsilon_{xx} & \varepsilon_{yy} & \varepsilon_{zz} & \gamma_{xy} & \gamma_{xz} & \gamma_{yz}\end{bmatrix}^T$$

is given by

$$\boldsymbol{\sigma} = \boldsymbol{D}(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon_0}), \tag{17}$$

where $\boldsymbol{D}$ is the constitutive matrix and $\boldsymbol{\varepsilon}_0$ is the initial strain. As there is plane strain, meaning $\varepsilon_{zz} = \gamma_{xz} = \gamma_{yz} = 0$, it follows that $\sigma_{xz} = \sigma_{yz} = 0$, and $\boldsymbol{D}$ and $\boldsymbol{\varepsilon}_0$ are consequently reduced in size. For an isotropic material,

$$\boldsymbol{D} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1}{2}(1-2\nu)\end{bmatrix} \tag{18}$$

and the initial strain is given by

$$\boldsymbol{\varepsilon_0} = (1+\nu)\alpha\Delta T \begin{bmatrix}1 & 1 & 0\end{bmatrix}^T. \tag{19}$$

The stress component σ_{zz} has to be calculated separately through

$$\sigma_{zz} = \frac{E\nu}{(1+\nu)(1-2\nu)}(\varepsilon_{xx} + \varepsilon_{yy}) - \frac{\alpha E \Delta T}{1-2\nu}. \tag{20}$$

The differential equation of equillibrium is given by

$$\tilde{\boldsymbol{\nabla}}\boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \tag{21}$$

which gives rise to the weak formulation,

$$\int_V (\tilde{\boldsymbol{\nabla}}\boldsymbol{v})^T \boldsymbol{\sigma}\, dV = \int_S \boldsymbol{v}^T \boldsymbol{t}\, dS + \int_V \boldsymbol{v}^T \boldsymbol{b}\, dV, \tag{22}$$

where $\boldsymbol{b}$ is the force per unit volume and $\boldsymbol{t}$ is the traction vector.

The nodal displacement vector is given by $\boldsymbol{u} = \boldsymbol{N}\boldsymbol{a}$, further meaning that $\boldsymbol{\varepsilon} = \tilde{\boldsymbol{\nabla}}\boldsymbol{u} = \boldsymbol{B}\boldsymbol{a}$. Using the Galerkin method, $\boldsymbol{v} = \boldsymbol{N}\boldsymbol{c}$, where $\boldsymbol{c}$ is arbitrary, and applying equation 17 results in

$$\left(\int_V \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}\, dV\right)\boldsymbol{a} = \int_S \boldsymbol{N}^T\boldsymbol{t}\, dS + \int_V \boldsymbol{N}^T\boldsymbol{b}\, dV + \int_V \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{\varepsilon_0}\, dV. \tag{23}$$

As the structure is a flat disc with thickness t, equation 23 may be reduced to its 2D equivalent,

$$\left(\int_S \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}t\, dS\right)\boldsymbol{a} = \oint_{\mathcal{L}} \boldsymbol{N}^T\boldsymbol{t}t\, d\mathcal{L} + \int_S \boldsymbol{N}^T\boldsymbol{b}t\, dS + \int_S \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{\varepsilon_0}t\, dS \tag{24}$$

$$\iff \boldsymbol{K}\boldsymbol{a} = \boldsymbol{f}_b + \boldsymbol{f}_l + \boldsymbol{f}_0. \tag{25}$$

On the top and bottom boundaries $\mathcal{L}_T$ and $\mathcal{L}_B$ it is known that $u_y = 0$, and on the rightmost boundary $\mathcal{L}_R$ $u_x = 0$. For the remaining components where the displacement is unknown, the traction is zero. Different properties hold for the different materials of the structure, meaning S should be separated into S_{Ti} for the titanium and S_{Gl} for the glass. $\boldsymbol{b} = \boldsymbol{0}$ as there are no forces acting upon the structure. Thus,

$$\begin{aligned}\left(\int_S \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}t\, dS\right)\boldsymbol{a} &= \int_{\mathcal{L}_T+\mathcal{L}_B} \boldsymbol{N}^T\boldsymbol{t}t\, d\mathcal{L} + \int_{\mathcal{L}_R} \boldsymbol{N}^T\boldsymbol{t}t\, d\mathcal{L} \\ &+ \int_{S_{Ti}} \boldsymbol{B}^T(\boldsymbol{D}\boldsymbol{\varepsilon_0})_{Ti}t\, dS + \int_{S_{Gl}} \boldsymbol{B}^T(\boldsymbol{D}\boldsymbol{\varepsilon_0})_{Gl}t\, dS,\end{aligned} \tag{26}$$

where appropriate values have been inserted into $\boldsymbol{a}$ and $\boldsymbol{t}$. As the known elements of $\boldsymbol{f}_b$ are 0, and the unknown elements aren't of interest, $\boldsymbol{f}_b$ may be ignored when solving the problem.

ΔT is in $\boldsymbol{f}_0$ (equation 25) integrated over the surface S, as $\boldsymbol{\varepsilon}_0$ is dependent of ΔT according to equation 19. As ΔT varies linearly between nodes, we may conclude that

$$\int_{S^e} \Delta T \, dS = S^e \cdot \frac{1}{3} \sum_{i=1}^{3} \Delta T_i \tag{27}$$

meaning the integral over an element surface may be reduced to the product of the surface area and the mean of the nodal ΔT.

The von Mises stress quantifies the total magnitude of the stress in a point and is defined as

$$\sigma_{eff} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 + \sigma_{xx}\sigma_{yy} + \sigma_{xx}\sigma_{zz} + \sigma_{yy}\sigma_{zz} + 3\tau_{xy}^2 + 3\tau_{xz}^2 + 3\tau_{yz}^2} \tag{28}$$

which functions as a measure of effective stress on the structure. Because of the plane strain conditions, equation 28 can be reduced to

$$\sigma_{eff} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 + \sigma_{xx}\sigma_{yy} + \sigma_{xx}\sigma_{zz} + \sigma_{yy}\sigma_{zz} + 3\tau_{xy}^2} \tag{29}$$

where σ_{zz} is given by equation 20.

2.3.1 MATLAB

The solution to the mechanical problem is presented in appendix A.5. ΔT is known for each node by running the solution to the stationary temperature problem (appendix A.3) setting T_∞ to either 20 °C or -96 °C depending on whether it is day- or nighttime. Then $\Delta T = T - 20$ °C.

The matrices $\boldsymbol{K}$ and $\boldsymbol{f}_0$ are generated using CALFEM's *plante.m* and *plantf.m* respectively, and equation 27 is used to calculate $\boldsymbol{D}\boldsymbol{\varepsilon}_0$. The nodal component displacement vector $\boldsymbol{a}$ is then calculated using CALFEM's *solveq.m*. The resulting $\boldsymbol{a}$ is then inserted as a parameter to CALFEM's *plants.m* to retrieve $\boldsymbol{\sigma}$, excluding the contribution from the initial strain $\boldsymbol{\varepsilon}_0$, which is later added according to equations 17, 18 and 19. As $\boldsymbol{\sigma}$ gives the stress of each element, the nodal von Mises stress is approximated as the mean of each connected element's von Mises stress.

2.4 Displacement analysis

To compare the magnitude of different distortions, the following quantity

$$\int_S \boldsymbol{u}^T \boldsymbol{u} t \, dS = \boldsymbol{a}^T \int_S \boldsymbol{N}^T \boldsymbol{N} t \, dS \cdot \boldsymbol{a}, \tag{30}$$

can be calculated, where $\boldsymbol{a}$ are the nodal component displacements in $\boldsymbol{u} = \boldsymbol{N}\boldsymbol{a}$, calculated in the the mechanical problem above..

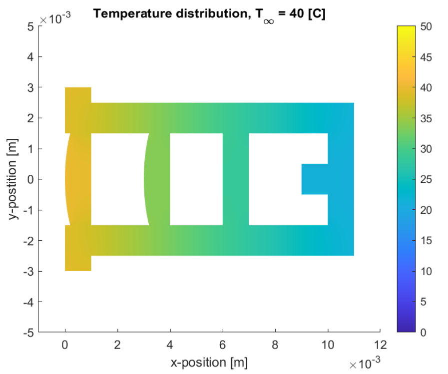
2.4.1 MATLAB

The solution is presented in appendix A.6. The nodal component displacements are retrieved by running the solution for the mechanical problem (appendix A.5) with either day- or nighttime conditions. Equation 30 is calculated for displacements that are within the subdomain of the leftmost lens and then summed. Furthermore, the displacement of the whole structure is plotted.

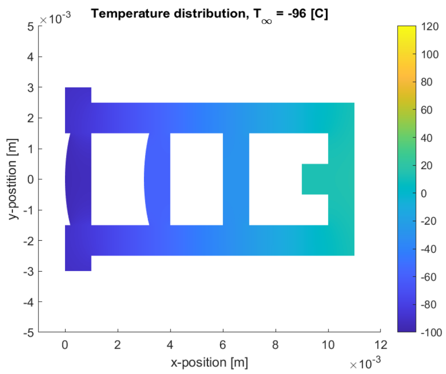
3 Results

3.1 Stationary temperature distribution

The resulting stationary temperature distributions for the day and night case are shown in figure 2. We see that the temperature in both cases changes gradually from one fixed temperature on one side to the other fixed temperature on the other side. There is no observable difference in behaviour between the titanium and glass materials.



(a) Stationary temperature distribution at daytime [°C]. Maximal calculated temperature: 39.954 °C

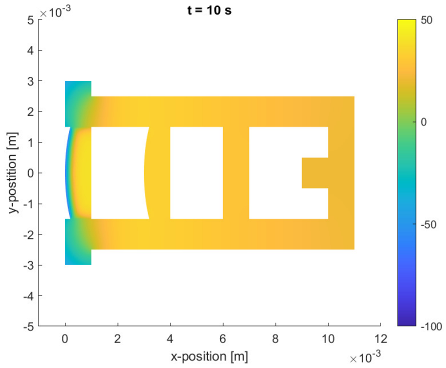


(b) Stationary temperature distribution at nighttime [°C]. Maximal calculated temperature: 19,9769 °C

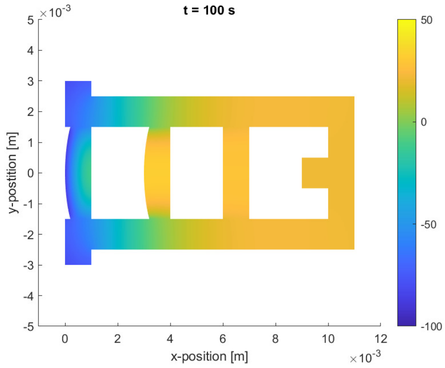
Figure 2: *Solutions to the stationary problem for different outside conditions. Note the different color scales.*

3.2 Transient temperature

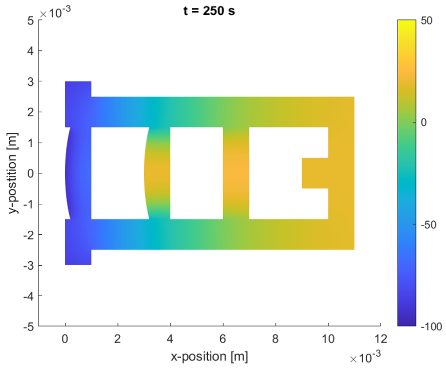
The results from when the boundary conditions switch from day to night conditions are shown in figure 3. The results from when the boundary conditions switch from night to day conditions are shown in figure 4. As expected, in both cases the new temperature slowly seeps in from the left and spreads through the system. Over time the temperature distribution more and more resembles the stationary solution. We can observe different behaviour in the titanium and the glass material. In the titanium the heat spreads faster and diffuses, causing a soft gradient whereas the heat in the glass spreads slower and thus has a sharper gradient. The difference can be observed most clearly in figure 3b and 4b.



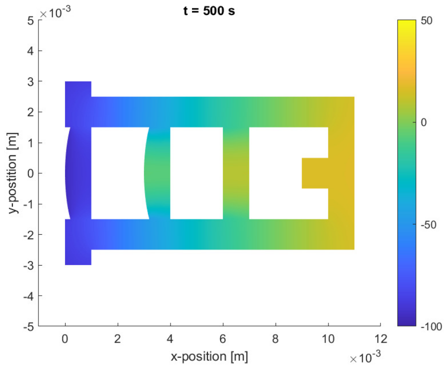
(a) Temperature distribution after 10 seconds [°C].



(b) Temperature distribution after 100 seconds [°C].

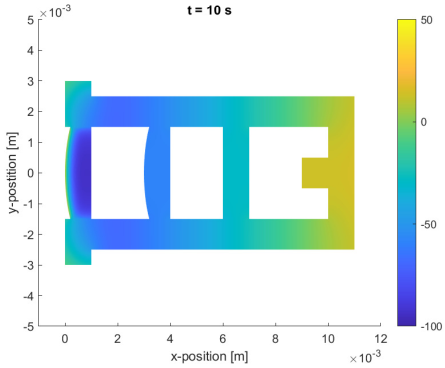


(c) Temperature distribution after 250 seconds [°C].

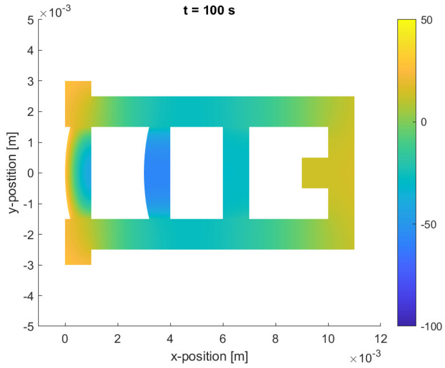


(d) Temperature distribution after 500 seconds [°C].

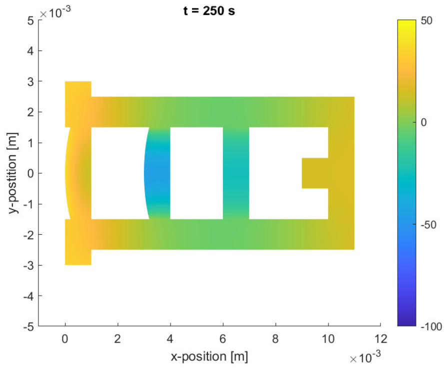
Figure 3: *Solutions to the transient heat problem, day to night.*



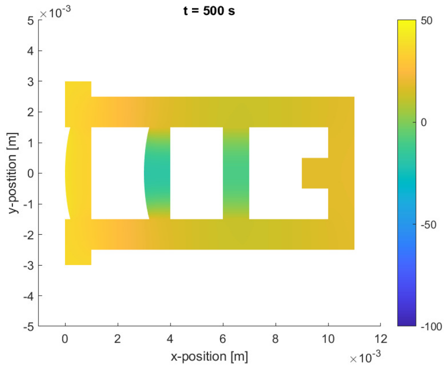
(a) Temperature distribution after 10 seconds [°C].



(b) Temperature distribution after 100 seconds [°C].



(c) Temperature distribution after 250 seconds [°C].

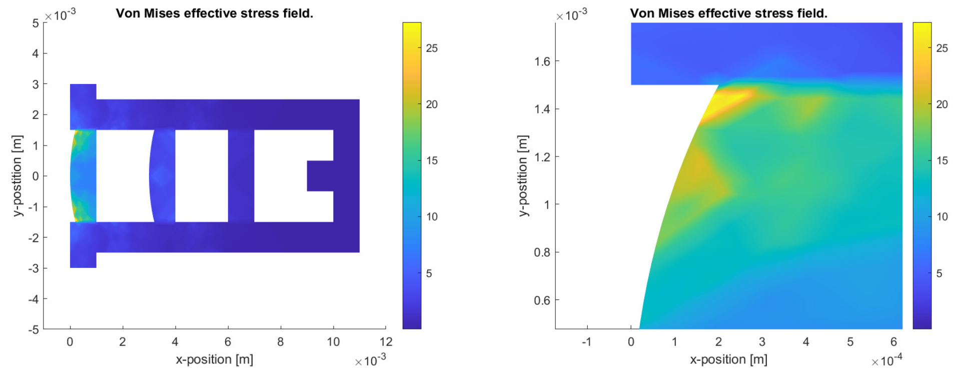


(d) Temperature distribution after 500 seconds [°C].

Figure 4: *Solutions to the transient heat problem, night to day.*

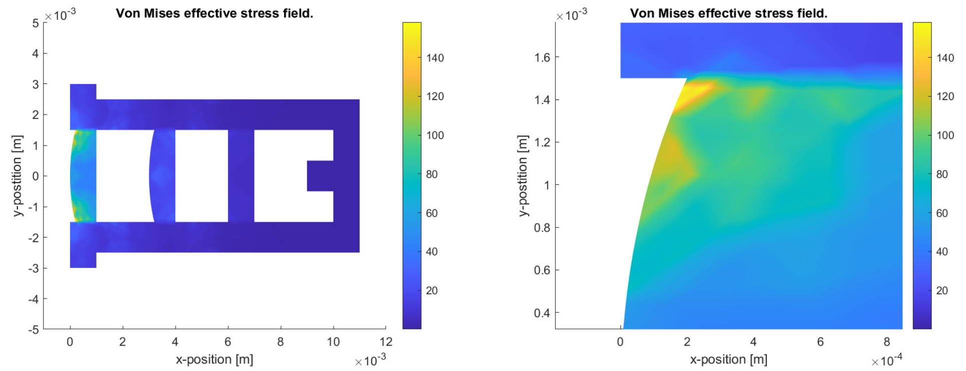
3.3 Body stresses

The von Mises effective stress in the system for the two different conditions is shown in figure 5 and 6. The stresses appear to be equally distributed but of different magnitudes, the night conditions yielding stresses about 6-7 times larger than the day conditions. This is expected as the conditions too only differ in magnitude, where the night temperature is about 6 times further from T_0 than the day temperature. Whether the relation is linear is unknown. The main stresses are found on the edges of the left lens, where there is a sharp edge and where the lens is constrained by the metal body. The detailed view in figure for example 5b shows that the stresses vary quickly between elements which could indicate that the model and geometry is sensitive to the mesh shape.



(a) Complete view. **(b)** Detailed view of top left lens-body border.

Figure 5: *The resulting von Mises effective stress field using the daytime condition [GPa].*



(a) Complete view. **(b)** Detailed view of top left lens-body border.

Figure 6: *The resulting von Mises effective stress field using the nighttime condition [GPa].*

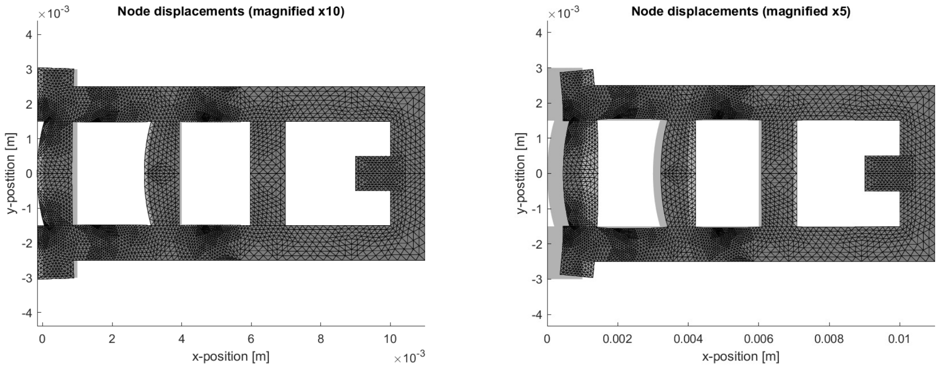
3.4 Node displacement

The calculated displacements for the day and night conditions are plotted in figure 18. As expected, the materials expand when heated and contract when cooled. The left side of the system is more distorted as it is closer to the boundary subjected to external heating. The titanium appears to distort more than the glass. We can also note that the night conditions cause a greater distortion which is due to being further from the stress free temperature.

This can be confirmed by integrating the displacement function over all elements in the the left lens as described in equation 30, yielding the following results:

- Day conditions: 3.2339e-18
- Night conditions: 1.0879e-16

meaning that the night conditions cause distortions about 30 times larger.



(a) Resulting nodal displacements using the daytime condition.

(b) Resulting nodal displacements using the nighttime condition.

Figure 7: *Magnified nodal displacements. The grey silhouette presents the displacement-free structure. Note the different magnifications.*

4 Discussion

In this report a two-dimensional thermo-elastic system modelling a lens system in a mars rover has been analyzed using the Finite Element Method. The stationary temperature distributions for day and night conditions as well as the transient temperature distributions when changing between the two were calculated. The calculated temperature distributions were then used to analyze the stress and distortion in the system when subjected to the different conditions. The von Mises stress field and the nodal displacements were calculated, as well as a quantity describing the total displacement in the left lens.

The investigation has found that the left lens, especially the area around the boundary between the lens and the metal body, is subjected to both stress and and distortion. The stresses are equally distributed but of different magnitudes for different temperatures. However, the materials could both expand and contract depending on the temperature. The right boundary of the lens bends significantly in night conditions, as shown in figure 7b, which could cause distortions in the image taken through the lens. The titanium body appears to distort more whereas stresses build up in the glass lenses. This is explained by the different material parameters and is in line with our intuitive understanding of the two materials. Heat also diffuses faster in the metal compared to the glass.

A limitation of the model is the assumption that the materials remain elastic at all times. Especially close to the boundary between the left glass lens and the titanium body, the stresses were quite high and could potentially reach plasticity. Further investigations of the behaviour of the materials are needed to conclude whether this has to be accounted for in the model.

To model the transient temperature more accurately, a continuous day-night has to be implemented. However, this would cause a slower temperature change which should allow the system to adapt slower. The abrupt switch between day and night used in this report could be used as a worst-case scenario when analyzing how the system reacts to changes in temperature.

A Computer code

A.1 Material data

```
% Material data relevant to the structure

TI = 1; % Titanium
GL = 0; % Glass

D = containers.Map({TI,GL},{eye(2)*17, eye(2)*0.8}); % Thermal
    conductivity [W/(m K)]
alpha_c = 100; % Heat transfer coefficient [W/(m^2K)]
T_0 = 20; % Temperature at which structure is stress free [C]

rho = containers.Map({TI,GL},{4620,3860}); % Density [kg/m^3]
E = containers.Map({TI,GL},{110, 67}); % Young's modulus [GPa]
alpha = containers.Map({TI,GL},{9.4e-6,7e-6}); % Expansion
    coefficient [1/K]
c_p = containers.Map({TI,GL},{523, 670}); % Specific heat [J/(
    kg K)]
Poisson = containers.Map({TI,GL},{0.34, 0.2}); % Poission's
    ratio [-]

thickness = 0.01; % Thickness of disc structure [m]
```

A.2 Geometry data

```
% Extract CALFEM notation from pdetool-mesh

% Element data
% Row #i of matrix refers to element #i

enod=t(1:3,:)'; % Nodes of elements
emat = (t(4,:)==TI)';  %  Material type of element
nelm = size(enod,1); % Number of elements

% Node data

coord = p'/100; % Coordinates of each node
nnod = size(coord,1); % Number of nodes
dof = (1:nnod)'; % Degrees of freedom
dof_S = [(1:nnod)',(nnod+1:2*nnod)']; % Give each dof a number
edof_S = zeros(nelm, 7);
```

```
edof = zeros(nelm, 4); %Element degrees of freedom

for ie = 1:nelm
    edof_S(ie,:) = [ie dof_S(enod(ie,1),:), dof_S(enod(ie,2),:)
        ,dof_S(enod(ie,3),:)];
    edof(ie,:) = [ie,enod(ie,:)];
end

% Define boundary conditions of element edges
% An edge of an element is defined by a node pair, [node #1;
    node #2]

er = e([1 2 5],:); % [edge; boundary segment]
conv_segments = [2 15 14 23 31 1]; % Boundary segments with
    convection
edges_conv = []; % Edges with convection

fixed_segments = [21 22 1 16 17 18 19 20]; % Fixed boundary
    segments
edges_fixed = []; % Fixed edges

for i = 1:size(er,2)
    if ismember(er(3,i),conv_segments)
        edges_conv = [edges_conv [er(1:2,i);er(3,i)==1]]; % [
            edge; convection type]
             % convection type = 0 -> T_inf outside
             % convection type = 1 -> T_c outside
    end
    if ismember(er(3,i),fixed_segments)
        edges_fixed = [edges_fixed [er(1:2,i);er(3,i)==1]]; % [
            edge; fixed type]
             %fixed type = 0 -> u_y = 0
             %fixed type = 1 -> u_x = 0
    end
end
```

A.3 Stationary temperature

```
% Stationary temporary problem

T_outside = [-96 20]; % Nighttime boundary temperatures [T_inf
    T_c]

```

```
K = zeros(nnod); % Global stiffness matrix
F = zeros(nnod,1); % Force vector

% Basic stiffness matrix

for ie = 1:nelm
    ex = coord(enod(ie,:),1)'; % x-coordinates of element
    ey = coord(enod(ie,:),2)'; % y-coordinates of element

    Ke = flw2te(ex,ey,thickness,D(emat(ie))); % Basic element stiffness matrix

    indx = edof(ie,2:end);  % Position in K
   K(indx,indx) = K(indx,indx)+Ke;  % Insert into K
end

% Add contribution to K and F from convection

fce_const = [1; 1]*alpha_c/2; % Common force vector integral constant
Kce_const = [2 1; 1 2]*alpha_c/6; % Common stiffness matrix integral constant

for ib = 1:length(edges_conv)
    indx = edges_conv(1:2,ib); % Edge with convection
    ex = coord(indx,1); % x-coordinates of nodes
    ey = coord(indx,2); % y-coordinates of nodes
    l = sqrt((ex(1)-ex(2))^2+(ey(1)-ey(2))^2);  % Edge length

    convectionTemp = edges_conv(3,ib)+1; % Nearby outside temperature
    fce = fce_const*l*T_outside(convectionTemp); % Force vector integral
    Kce = Kce_const*l; % Stiffness matrix integral

   K(indx,indx) = K(indx,indx)+Kce;  % Insert into K
    F(indx) = F(indx) + fce; % Insert into F
end

% Solve system

a0 = solveq(K,F); % Nodal temperatures
[ex, ey] = coordxtr(edof, coord, dof, 3); % Extract nodal
```

```
    coordinate data
ed = extract(edof, a0); % Extract element temperatures

% Plot

clf
patch(ex',ey',ed','EdgeColor','none');
hold on
patch(ex',-ey',ed','EdgeColor','none');

caxis([-100 50]);
axis([-0.1 1.2 -0.5 0.5]/100);
title("Temperature distribution, T_{\infty} = " + (T_outside(1)) + " [C]");
%colormap(hot);
colorbar;
xlabel('x-position [m]')
ylabel('y-postition [m]');

disp("MAX TEMP: " + max(max(ed)));

% Max T vid T_inf = -96: 19,9769
% Max T vid T_inf = 40: 40,0093
```

A.4 Transient heat

```
% Transient heat problem

delta_t = 100; % Time step length
T_outside = [-96 20]; % Initial boundary temperatures [T_inf T_c]
a = a0; % Initial nodal temperatures, generated in a)

A = zeros(nnod); % Damping matrix
F = zeros(nnod,1); % Force vector

% Generate F with initial boundary temperatures

fce_const = [1; 1]*alpha_c/2;
for ib = 1:length(edges_conv)
    indx = edges_conv(1:2,ib); % Edge with convection
    ex = coord(indx,1); % x-coordinates of nodes
    ey = coord(indx,2); % y-coordinates of nodes
```

```
    l = sqrt((ex(1)-ex(2))^2+(ey(1)-ey(2))^2);  % Edge length

    convectionTemp = edges_conv(3,ib)+1; % Nearby outside temperature
    fce = fce_const*l*T_outside(convectionTemp); % Force vector integral
    F(indx) = F(indx) + fce; % Insert into F
end

% Damping matrix A

for ie = 1:nelm
    ex = coord(enod(ie,:),1); % x-coordinates of nodes
    ey = coord(enod(ie,:),2); % y-coordinates of nodes

    material = emat(ie); % Material type of element
    Ae = plantml(ex',ey',rho(material)*c_p(material)); % Element damping matrix

    indx = edof(ie,2:end);  % Position in A
    A(indx, indx) = A(indx,indx)+Ae; % Insert into A
end

% Implicit Euler time step function

time_step = @(a) (A + delta_t*K)\(F*delta_t+A*a);

% Temperature development over time

for i=1:200 % 200 time steps
    a = time_step(a);
end

[ex, ey] = coordxtr(edof, coord, dof, 3); % Extract nodal coordinate data
ed = extract(edof, a); % Extract element temperatures

%Plot

patch(ex',ey',ed','EdgeColor','none');
hold on
patch(ex',-ey',ed','EdgeColor','none');

```

```
caxis([-100 50]);
axis([-0.1 1.2 -0.5 0.5]/100);
title("t = " + (i*delta_t) + " s");
colormap default;
colorbar;
xlabel('x-position [m]');
ylabel('y-postition [m]');

disp("MAX TEMP: " + max(max(ed)));
disp("MIN TEMP: " + min(min(ed)));
```

A.5 Displacements and von Mises stress

```
% Mechanical problem

% D matrix for isotropic material with plane strain

calcD = @(E, v) E/(1+v)*(1-2*v)*[(1-v) v 0; v (1-v) 0; 0 0
    (1-2*v)/2];
D_el = containers.Map({TI,GL},{calcD(E(TI),Poisson(TI)),calcD(E
    (TI),Poisson(GL))});

% Define constant part of D*epsilon_0

calcConstEps_0 = @(mat) alpha(mat)*E(mat)/(1-2*Poisson(mat))
    *[1;1;0];
const_Deps0 = containers.Map({TI,GL},{calcConstEps_0(TI),
    calcConstEps_0(GL)});

K = zeros(nnod*2); % Global stiffness matrix
F = zeros(nnod*2,1); % Force vector

for ie = 1:nelm
    ex = coord(enod(ie,:),1)'; % x-coordinates of nodes
    ey = coord(enod(ie,:),2)'; % y-coordinates of nodes
    material = emat(ie); % Material type of element

    Ke = plante(ex, ey, [2 thickness], D_el(material)); %
        Element stiffness matrix

    dT = mean(ed(ie,:))-T_0; % Mean temperature in element
    es = const_Deps0(material)*dT; % Element D*epsilon_0
    f_0e = plantf(ex, ey, [2 thickness], es'); % Element f_0
```

```
    % (f_be = 0)

    indx = edof_S(ie,2:end);  % Position in matrix
    K(indx, indx) = K(indx,indx)+Ke; % Insert into K
    F(indx) = F(indx) + f_0e; % Insert into F
end

% Calculate bc

already_added = zeros(nnod*2,1); % Memory vector
bc = []; % Boundrary condition vector

for ib = 1:length(edges_fixed)
    edge = edges_fixed(:,ib); % [edge; fixed type]
    x_or_y = 2-edge(3); % Fixed type

    %Check first node

    node_id = dof_S(edge(1),x_or_y); % Index of fixed node component

    if(already_added(node_id)==0) % if it hasn't been added...
        bc = [bc; node_id, 0]; % Add to bc
        already_added(node_id) = 1; % Update memory vector
    end

    % Check second node

    node_id = dof_S(edge(2),x_or_y); % Index of fixed node component

    if(already_added(node_id)==0) % if it hasn't been added...
        bc = [bc; node_id, 0]; % Add to bc
        already_added(node_id) = 1; % Update memory vector
    end
end

% Solve system

a_S = solveq(K,F, bc); % Nodal component displacements
[ex_S, ey_S] = coordxtr(edof_S, coord, dof_S, 3); % Extract nodal coordinate data
ed_S = extract(edof_S, a_S); % Extract element displacements
```

```
% Calculate von Mises stress per element

Seff_el = zeros(nelm,1); % Element von Mises stress

for ie = 1: nelm
    ex = coord(enod(ie,:),1)'; % x-coordinates of nodes
    ey = coord(enod(ie,:),2)'; % y-coordinates of nodes
    material = emat(ie); % Material type of element
    dT = mean(ed(ie,:))-T_0; % Mean temperature in element

    a_index = [dof_S(enod(ie,:), 1) ; dof_S(enod(ie,:), 2)]; % Nodal component indices in a_S

    % [sigma_xx sigma_yy sigma_xy]
    sigma1 = plants(ex, ey, [2 thickness], D_el(material),a_S(a_index)'); % No initial strain
    sigma = sigma1 - (const_Deps0(material)*dT)'; % Add contribution from initial strain

    % sigma_zz
    sigma_zz1 = Poisson(material)*(sigma(1) + sigma(2)); % No initial strain
    sigma_zz = sigma_zz1 - alpha(material)*E(material)*dT/(1-2*Poisson(material)); % Add contribution from initial strain

    vonMisesSquared = sigma*sigma' + sigma_zz^2 - sigma(1)*sigma(2)-sigma(1)*sigma_zz-sigma(2)*sigma_zz+2*sigma(3)^2;
    Seff_el(ie) = sqrt(vonMisesSquared);
end

% Calculate nodal von Mieses stress as mean of connected elements

Seff_nod = zeros(nnod,1); % Nodal von Mises stress

for i=1:nnod
    [c0, c1] = find(edof(:,2:4)==i); % Row indices of connected elements
    Seff_nod(i,1) = sum(Seff_el(c0))/size(c0,1); % Mean von Mises stress
```

```
end

eM = extract(edof, Seff_nod); % Extract nodal von Mises stress

% Plot

patch(ex_S',ey_S',eM','EdgeColor','none');
hold on
patch(ex_S',-ey_S',eM','EdgeColor','none');

axis([-0.1 1.2 -0.5 0.5]/100);
title("Von Mises effective stress field.");
colormap default;
colorbar;
xlabel('x-position [m]');
ylabel('y-postition [m]');
```

A.6 Displacement field

```
lens_subdomain = 3;
lens_displacement = 0; % Total lens displacement

for ie = 1:nelm
    if(t(4,ie)==lens_subdomain) % if element is within lens subdomain...
        ex = coord(enod(ie,:),1)'; % x-coordinates of nodes
        ey = coord(enod(ie,:),2)'; % y-coordinates of nodes

        T = NtN(ex, ey,thickness); % Compute int(N^T*N)t dA
        a_x = a_S(edof_S(ie,2:4)); % Nodal x-component displacement
        a_y = a_S(edof_S(ie,5:7)); % Nodal y-component displacement
        a = [a_x; a_y]; % Nodal displacement

        lens_displacement = lens_displacement + a'*T*a; % Compute lens displacement and add
    end
end

disp("TOTAL LENS DISPLACEMENT: " + lens_displacement);

% Plot
```

```
mag = 10; % Displacement magnification
exd = ex_S + mag*ed_S(:,1:2:end); % Nodal x-coordinate data
eyd = ey_S + mag*ed_S(:,2:2:end); % Nodal y-coordinate data

figure();
patch(ex_S',ey_S',[0 0 0],"EdgeColor","none","FaceAlpha",0.3);
hold on
patch(ex_S',-ey_S',[0 0 0],"EdgeColor","none","FaceAlpha",0.3);
patch(exd',eyd',[0 0 0],"FaceAlpha",0.3);
patch(exd',-eyd',[0 0 0],"FaceAlpha",0.3);
axis equal

xlabel('x-position [m]');
ylabel('y-postition [m]');
title("Node displacements (magnified x10)");
```

A.6.1 Integral computation function

```
function Te=NtN(ex,ey,t)
% T=(ex,ey,t)
%-------------------------------------------------------------
% PURPOSE
%  Compute the quantity: Te=int(N^T*N)t dA
%
% INPUT:   ex,ey;        Element coordinates
%          t;            Element thickness
%
%
% OUTPUT: Te :        Matrix 3 x 3
%-------------------------------------------------------------

Area=1/2*det([ones(3,1) ex' ey']);

L1=[0.5 0 0.5];
L2=[0.5 0.5 0];
L3=[0 0.5 0.5];

NtN=zeros(6);


for i=1:3
        NtN=NtN+1/3*[L1(i)^2 0 L1(i)*L2(i) 0 L1(i)*L2(i) 0
```

```
                                                    0 L1(i)^2 0 L1(i)*L2(i) 0 L1(i)
                                                        *L2(i)
                                                    L2(i)*L1(i) 0 L2(i)^2 0 L2(i)*
                                                        L3(i) 0
                                                    0 L2(i)*L1(i) 0 L2(i)^2 0 L2(i)
                                                        *L3(i)
                                                    L3(i)*L1(i) 0 L3(i)*L2(i) 0 L3(
                                                        i)^2 0
                                                    0 L3(i)*L1(i) 0 L3(i)*L2(i) 0
                                                        L3(i)^2];
end

Te=NtN*Area*t;
```